EVENT LOOP AND Concurrency in JavaScript

01

WHAT IS THE EVENT LOOP?

The Event Loop is a mechanism that allows JavaScript to perform non-blocking I/O operations despite being single-threaded.

02

CALL STACK

JavaScript uses the Call Stack to manage function execution. Functions are pushed and popped in a LIFO (Last In, First Out) order.

```
function foo() {
  console.log('foo');
}
foo(); // 'foo'
```

03

WEB APIS

Web APIs handle asynchronous tasks like HTTP requests, setTimeout, and event listeners, moving them outside the Call Stack.

```
setTimeout(() => console.log('Hello'), 1000);
```

04

CALLBACK QUEUE

When async operations complete, their callbacks are placed in the Callback Queue, waiting to be pushed onto the Call Stack.

05

EVENT LOOP IN ACTION

The Event Loop continuously checks if the Call Stack is empty. If it is, it pushes the first callback from the Callback Queue to the Call Stack.

```
console.log('Start');
setTimeout(() => console.log('Middle'), 0);
console.log('End');
// Output: 'Start', 'End', 'Middle'
```

06

CONCURRENCY

JavaScript achieves concurrency by leveraging the Event Loop, allowing it to handle multiple tasks efficiently without multiple threads.

08

SUMMARY

The Event Loop enables JavaScript's single-threaded concurrency, managing async tasks through the Call Stack, Web APIs, and Callback Queue for smooth execution.

FOLLOW

FOR MORE